



Élaboration d'une aide à la décision

Dimitri Dupuis-Latour — Hôpital Européen Georges Pompidou

Rapport de Stage FICM 2A — 26 Juin / 8 Septembre 2006

Table des matières

Introduction

Genèse de mon projet de stage

L'Hôpital Européen Georges Pompidou (HEGP) recherchait, dans le cadre d'un stage d'été de 10 semaines, un élève ingénieur pour développer des outils d'aide à la prescription médicale. Ce sujet de stage m'intéressa particulièrement du fait qu'il mêlait informatique pure et besoins clients : programmation, structure de données et écoute / synthèse du besoin client ; c'est précisément ce à quoi j'ai été formé à l'école, ayant choisi la filière MOSI : "Maîtrise d'Ouvrage" des Systèmes d'Information. De plus, j'aime innover, inventer, trouver des solutions nouvelles. Ce stage proposait justement d'élaborer une solution adaptée, originale, pour des utilisateurs particuliers : les médecins.

Descriptif du Stage

Le stage concernait la prescription de certains médicaments anticoagulants oraux : les anti-vitamine K (AVK), utilisés dans le traitement de maladies cardio-vasculaires et la prévention de la thrombose (encore appelée phlébite). La posologie de ces médicaments est variable : elle doit être ajustée en fonction d'un test biologique, l'INR (International Normalized Ratio), qui reflète le degré de coagulabilité du sang du patient. Mon stage a consisté à mettre en place un outil d'aide à la prescription de ces médicaments. Les différentes étapes de ce projet impliquaient des phases d'analyse, de recherche, de modélisation, de développement, de documentation et de présentation des travaux.

Plan du rapport

Je présenterai d'abord le cadre dans lequel j'ai travaillé : le milieu médical, l'hôpital, et son système d'information ; puis les deux projets (1 & 2) qui constituaient mon stage, expliquant à chaque fois l'objectif initial, les résultats obtenus, et les perspectives ouvertes, puis dresserai le bilan de ce que le stage m'a apporté.



Ce document a été rédigé à destination de deux types d'audience : les professeurs de l'École des Mines, et l'équipe évaluation de l'HEGP. Afin de séparer les aspects organisationnels et techniques du stage, ce rapport a été séparé en deux parties : le corps du document propose une présentation générale du stage ; une annexe permet quant à elle de recueillir les aspects plus techniques et spécifiques.

Remerciements

J'adresse tous mes remerciements à Éric Zapletal (ingénieur de recherche), Julie Niès (ingénieur thésard) et Isabelle Colombet (encadrement médical), de l'équipe Évaluation du DIH, qui m'ont accueilli et consacré une part importante de leur temps à répondre à mes questions.

Contact :
Hôpital Européen Georges Pompidou
Direction de l'Informatique Hospitalière
15 rue Leblanc — 75015 PARIS
01 56 09 20 00 postes 17 252 & 17253

eric.zapletal@egp.aphp.fr
julie.nies@spim.jussieu.fr

Cadre du Stage

Présentation de l'Hôpital

Historique

L'Hôpital Européen Georges Pompidou (HEGP), situé dans le 15e arrondissement de Paris, est le dernier né des grands hôpitaux parisiens de l'Assistance Publique-Hôpitaux de Paris (AP-HP). Mis en service en l'an 2000, l'HEGP remplace les hôpitaux Boucicaut (Paris 15ᵉ), Broussais (Paris 14ᵉ) et Laennec (Paris 7ᵉ), maintenant fermés. Le bâtiment, moderne et fonctionnel, est organisé autour de 4 pôles reliés entre eux par une rue hospitalière piétonne couverte (photo de couverture), qui facilite ainsi l'organisation de l'espace.

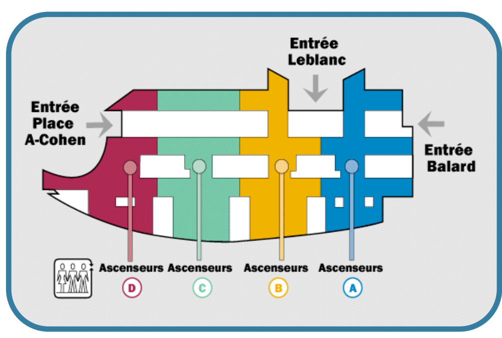


Figure 1 : Les 4 pôles de l'Hôpital

Missions

L'HEGP répond à une double vocation : il assure d'une part, une mission de proximité en réponse aux besoins de santé de la population adulte du Sud-Ouest parisien (7ᵉ, 15ᵉ, Issy-les-Moulineaux, Malakoff, Montrouge, Vanves…) ; d'autre part, il assure des soins d'hospitalisation aiguë dans trois pôles d'activité : pôle cardio-vasculaire, cancérologie & spécialités médico-chirugicales, et urgences, disposant pour cela d'une capacité d'accueil de près de 900 lits.

L'HEGP, en tant qu'hôpital universitaire, alloue également d'importants moyens à la recherche : lui sont rattachées 8 unités INSERM, 2 unités CNRS, des laboratoires universitaires, ainsi que d'autres unités de recherche clinique (Investigation, Épidémiologie…). Un bâtiment de recherche de 4500 m^2, actuellement en construction, aura d'ailleurs pour objectif de développer sur site des activités de recherche étroitement associées à l'activité médicale et de rassembler en un site unique des structures aujourd'hui dispersées.

Fonctionnement informatique

L'HEGP, de par sa récente mise en service, bénéficie d'un système de soins entièrement informatisé [1] : gestion des lits ; prescription de médicaments, d'examens de biologie et d'imagerie ; visualisation des résultats, expérimentation du dossier médical personnel informatisé (carnet de santé numérique)… Les applications métiers au cœur de l'activité de production de soins, coordonnées par le logiciel DxCare®, permettent ainsi d'effectuer un circuit de prescription complet dans l'hôpital.

Sept cents ordinateurs sont utilisés pour la production de soins (en unités de soins et dans les bureaux médicaux) ; cet équipement comprend notamment des PC portables spéciaux (stérilisables), permettant l'accès au système d'information au chevet du patient.



Figure 2 : Accès au Système d'Information au chevet du patient

Présentation du Service

Le DIH

Afin de gérer son Système d'Information (SI), l'hôpital s'est doté d'un Département d'Informatique Hospitalière (DIH), qui en gère les aspects techniques et applicatifs :

- Technique : gestion de l'infrastructure et des réseaux, installation et déploiement, configuration, support aux utilisateurs.
- Applicatif : gestion des différentes composantes du SI (gestion des versions, mises à jour, relations avec les industriels).

L'équipe "Évaluation & Gestion des Connaissances"

Au sein du DIH, l'équipe d'évaluation des pratiques médicales, dirigée par le Dr Pierre Durieux, a pour mission d'évaluer et d'améliorer la qualité des pratiques médicales, en s'appuyant sur le SI pour deux types d'actions :

- d'une part la mise en place de systèmes d'aides intervenant au moment de la prescription (Aide à la décision), pour conseiller le médecin. L'affichage d'une telle aide est figurée ci-contre.
- d'autre part le recueil en routine de critères d'évaluation des pratiques professionnelles (Monitoring & QA : Quality Assurance).

L'équipe évaluation recueille et analyse ces données. Leur exploitation pour l'évaluation des pratiques de soins, si elle est possible et validée, représente l'avantage de pérenniser un recueil régulier de données, et permet d'envisager dans la routine la mise en œuvre et l'évaluation d'actions d'amélioration des pratiques de soins.

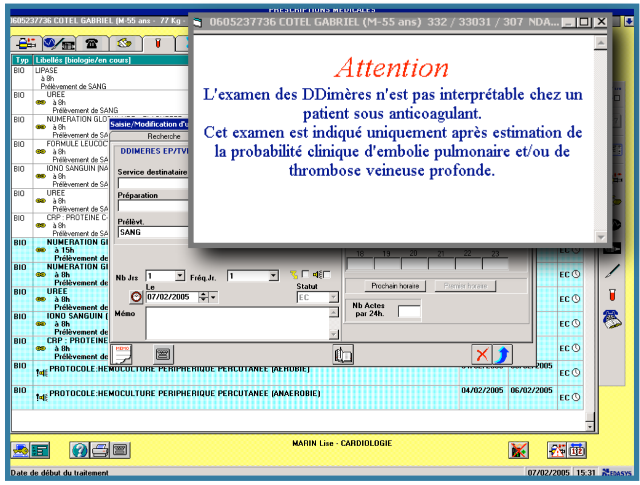


Figure 3 : Affichage d'une aide pendant la prescription

Contexte du projet

Double besoin

D'après une enquête [2] conduite en France par les centres de pharmacovigilance, les accidents hémorragiques accompagnant un traitement par anticoagulant constituent en France une des premières causes d'hospitalisation pour effet indésirable médicamenteux. Sur la base de ce constat, l'AFSSAPS (Agence Française de Sécurité Sanitaire de Produits de Santé) a fait de la prévention du surdosage aux anticoagulants AVK une de ses priorités.

Objectifs initiaux

L'affichage d'une aide au moment de la prescription est susceptible de constituer une mesure permettant d'améliorer les pratiques de prescription et de diminuer la fréquence des effets indésirables. Il est ainsi possible, à partir de nomogrammes établis pour l'ajustement des traitements par AVK [3], d'envisager une aide automatisée intégrée au système informatique de prescription de médicaments. *La première partie de mon stage a consisté à mettre en œuvre cette aide pour les AVK.*

D'autre part, la mise en œuvre d'un nombre croissant d'aides, tous médicaments confondus, a conduit le DIH à envisager l'adoption d'un outil de création simplifié et automatisé de ces aides, tel qu'un moteur de règles ; mais il reste encore à établir et documenter l'intérêt d'investir dans cet outil. *La seconde partie de mon stage a donc consisté à évaluer cet outil, et à en dégager un intérêt éventuel quant à l'édition de règles pour le DIH.*

Présentation de la mission

Au sein de l'équipe évaluation du DIH, mon stage avait pour double objectif de générer une aide pour les AVK en employant la méthode traditionnelle du service (langage Javascript), puis en utilisant un moteur de règles, afin de pouvoir montrer :

I. la faisabilité d'une aide AVK dans le système d'information de l'HEGP ;
II. la simplification de cette mise en œuvre grâce à l'utilisation d'un moteur de règles.

1ère Partie : Projet Javascript

Objectif Initial

La première mission qui me fut confiée consista à mettre en œuvre l'affichage d'une aide automatisée pour les AVK, intégrée au système informatique de prescription de médicaments, que je réalisa en 7 étapes :

1. Compréhension du problème médical & analyse des spécifications fonctionnelles ;
2. Traduction, en algorithme, d'un tableau d'ajustement de la posologie des AVK validé cliniquement ;
3. Évaluation & recherches d'outils de développement adaptés pour l'édition de code et la gestion de projets ; Apprentissage de CSS & Javascript ;
4. Développement d'un prototype en local sur mon ordinateur (sans appel à DxCare®) ;
5. Réalisation d'une batterie de tests, en m'appuyant sur une base de patients fictifs, élaborée par mes soins ;
6. Intégration au sein du logiciel DxCare®, utilisé par les médecins pour toute prescription ;
7. Mise en production dans l'hôpital (services de médecine vasculaire et de médecine interne)
8. Génération d'une documentation technique sur le travail effectué, afin de faciliter la maintenance du code réalisé.

Résultats

La compréhension du problème médical (étapes 1 & 2) consista à comprendre et à formaliser les spécifications médicales, contenues dans un document de référence sur lequel je me suis basé. Ces deux étapes se sont poursuivies en réalité pendant les étapes 4 & 5, car les spécifications étaient parfois ambiguës (lors de cas particuliers non envisagés), parfois changeantes (par ex. : idée d'afficher un diagramme est seulement apparue suite à l'étape 5). De plus, le document de spécification datait de 3 ans, et concernait un projet plus ambitieux : il fallait donc parfois savoir le ré-interpréter.

La recherche d'outils de développement (étape 3) a permis d'évaluer différents outils pour éditer le code (HTML, CSS et Javascript) ; ainsi que différents systèmes de gestion de versions : CVS, Trac, et notamment Subversion. Les outils retenus, essentiellement Xcode et Safari, sont présentés Figure 4, ainsi que les alternatives existant sur les autres plateformes.

<table>
<tr><th>Langage</th><th>Référence</th><th>Éditeur</th><th>Validateur
(Vérification statique)</th><th>Débogueur
(Vérification dynamique)</th></tr>
<tr><td>HTML
(Structure)</td><td>HTML 4.01</td><td rowspan="3">Xcode
Éclipse WTP</td><td>HTML Tidy (plug-in Safari)
HTML Tidy (plug-in Firefox)
HTML Validator (on-line)</td><td rowspan="2">Safari (Web Inspector)
Firefox (Inspecteur DOM)</td></tr>
<tr><td>CSS
(Présentation)</td><td>CSS 2.1</td><td>CSS Validator (on-line)
CSS Validator (on-line)</td></tr>
<tr><td>Javascript
(Comportement)</td><td>W3 School</td><td>Firefox (Javascript Console)
Firefox (Javascript Console)</td><td>Safari (Drosera)
Firefox (Venkman Debugger)</td></tr>
<tr><td colspan="3">Étapes : Code écrit</td><td>Code grammaticalement correct</td><td>Code fonctionnel</td></tr>
</table>

Figure 4 : Tableau récapitulatif des outils de développements utilisés.
En Souligné : outils utilisés (Environnement Mac OS X). En Italique : alternatives possibles (Autres environnements).

Cette étape fut également consacrée à l'apprentissage de HTML, CSS et Javascript, même si celui-ci a été effectué en parallèle avec l'écriture du code. CSS et HTML sont des langages précis et bien documentés ; je les ai utilisés en m'aidant des spécifications éditée par le W3C, claires et concises [4] [5]. JavaScript, en revanche, n'est pas un standard : il ne bénéficie pas d'une documentation de référence ; sa syntaxe diffère selon les navigateurs, et les sites web qui tentent de l'expliquer sont souvent obsolètes (pré-2000, ne mentionnant que Netscape (v4) et Internet Explorer (v5)). Une documentation de qualité fut longue à trouver, mais existe [6] ; les incompatibilités entre navigateurs restent cependant toujours présentes [7].

L'écriture et le test du code (étapes 4 & 5), plus techniques, sont détaillées dans l'annexe "Guide de l'Aide AVK". Ils consistaient à écrire une page HTML, qui embarquait un programme Javascript, lequel appelait des scripts PHP effectuant des requêtes SQL (Cf. Figure 5). La validation du code en cours d'écriture, fort pratique, est rendue possible par des outils adaptés [8][9][10].

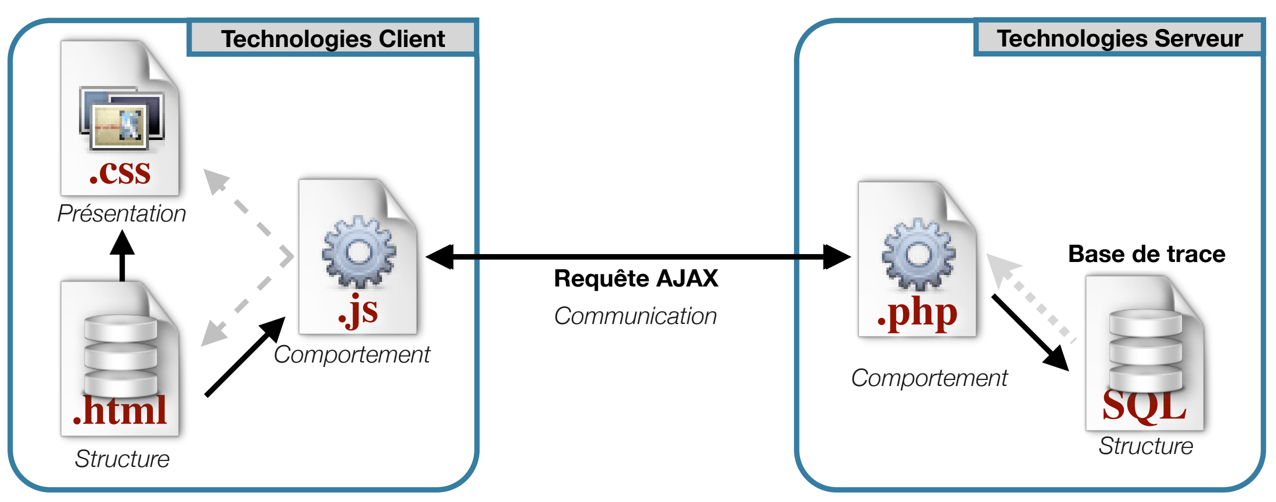


Figure 5 : Schéma de principe de l'Aide AVK, illustrant les différents technologies mises en jeu.

Puis, afin de faciliter les tests du code, j'ai créé une interface de débogage HTML qui affichait en plus de l'aide, l'état des variables du code (Figure 6). L'interface ①, destinée au développeur, a par la suite été retirée de la version finale. J'ai ensuite créé une base de tests avec des patients fictifs, chacun représentant un stade de traitement dans le nomogramme de référence [3], et représenté par un numéro ③. Ceci a permis d'une part d'automatiser les tests (la sélection d'un seul patient initialise le test avec toutes les variables correspondantes), d'autre part de "capitaliser" le temps passé à l'élaboration de ces bases, cette réutilisation se traduisant par d'importants gains de temps.

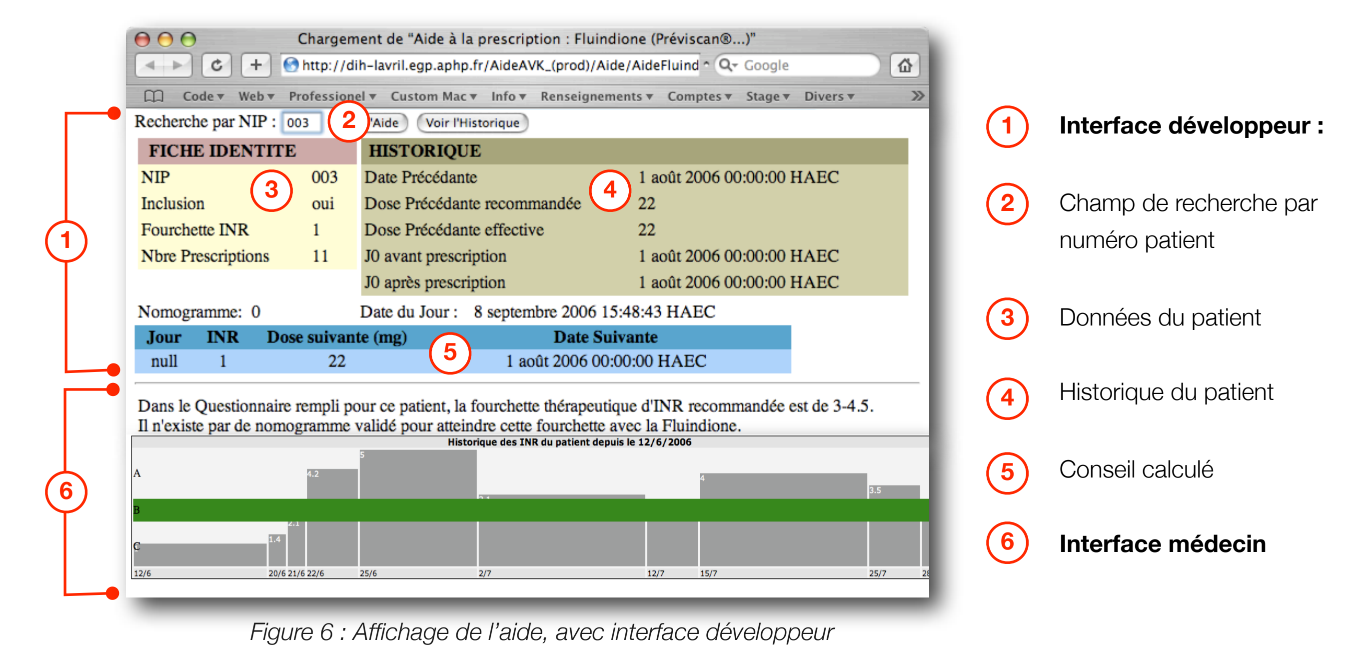


Figure 6 : Affichage de l'aide, avec interface développeur

Enfin, au cours du développement de cette aide, j'ai proposé d'améliorer la présentation de l'information (Figure 7) en ajoutant, en plus de la dose conseillée ①, un histogramme ②. Ce diagramme, non prévu initialement, visait à mieux faire accepter aux médecins la dose conseillée par le logiciel, en montrant graphiquement l'évolution de l'état du patient. Le rendu visuel permet aussi de comprendre l'information plus rapidement, notamment grâce à un code de couleurs (vert, orange ou rouge, selon le risque).

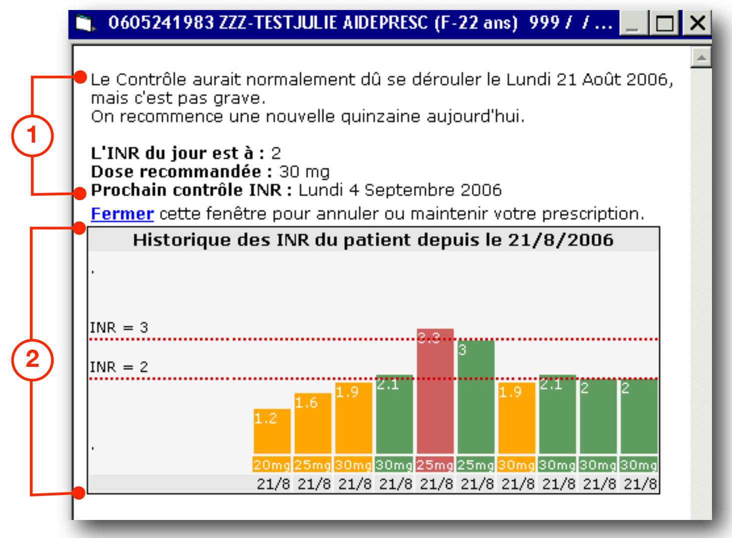


Figure 7 : Affichage de l'aide, avec conseil et diagramme

Ce diagramme n'ayant pas été spécifié formellement (car proposé par mes soins), j'ai dû réaliser plusieurs prototypes d'affichage (Figure 8), évalués à chaque fois par les médecins prescripteurs, et améliorés successivement grâce à leurs commentaires. C'est ainsi que l'interface finale ② gagna l'affichage des doses par rapport à la version ①; la coloration des barres par rapport à la version ③, ainsi que l'affichage de la fourchette INR 3 à 4,5 le cas échéant (Dans ce cas, les barres d'INR ont une coloration neutre : gris foncé). L'échelle des temps a également changé de comportement : elle fut d'abord proportionnelle ③, puis à écartement constant (*i.e.* barres verticales à largeur fixe, comme en ① et ②). Dans une version intermédiaire non figurée ici, elle fut aussi "proportionnelle avec zoom automatique" (*i.e.* l'échelle des temps était automatiquement ajustée pour faire tenir l'historique des 14 derniers résultats INR disponibles dans la largeur de la fenêtre, et non dans un long ruban comme en ③).

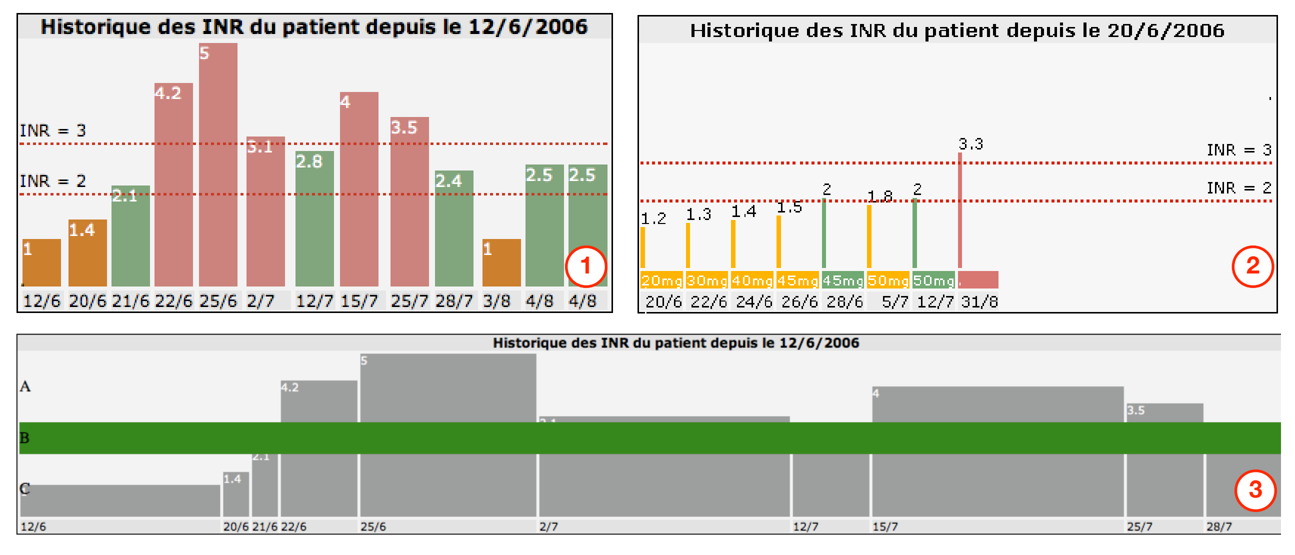


Figure 8 : Prototypes d'affichage de l'aide.

L’intégration avec DxCare® (étape 6), fut longue et fastidieuse, pour deux raisons : tout d’abord, l’architecture de déploiement, complexe (DxCare®, la page HTML, la base SQL, la simulation d’un traitement sur plusieurs jours…), ne permettait pas d’automatiser les tests, comme cela a pu être fait en local (Étape 4-5). Ainsi, entre deux éditions successives du code, il fallait manuellement re-paramètrer l’aide, vider la base, prescrire, analyser l’erreur, etc.

D’autre part, l’utilisation d’IE 5 comme interface de visualisation compliqua aussi le débogage : en plus du non-respect de certains standards empêchant l’emploi de fonctionnalités Javascript et CSS avancées, ce navigateur ne dispose pas, dans sa version intégrée à DxCare®, d’outils de débogage permettant d’inspecter, dynamiquement, l’état des variables, la position dans le Javascript, ou le code HTML généré par le Javascript. Il retourne simplement la ligne où il s’est arrêté, à la suite de quoi il faut essayer d’imaginer d’où vient l’erreur, essayer une autre solution, puis recommencer un cycle de tests complet, sans garantie d’avoir identifié précisément l’erreur. Il faut donc en général faire plusieurs de ces cycles pour localiser et corriger chaque erreur, ce qui est assez fastidieux. Heureusement, une part importante du code avait déjà été débogué en local, avec des outils plus adaptés [11][12].

Mise en production (étape 7). Après avoir été testé auprès des médecins du DIH et des pharmaciens, il reste encore à effectuer la mise en production de l’aide, dans les services de médecine vasculaire et de médecine interne. Débutera alors une étude d’environ un an, qui visera à évaluer l’impact de cette aide sur les pratiques médicales.

La documentation technique générée (étape 8) est constituée du présent chapitre ainsi que de l'annexe de fin de document qui recueille pour sa part les aspects les plus techniques et spécifiques.

Perspectives

S’il se confirme que la visualisation sous forme d’histogrammes apporte une valeur ajoutée au médecin, ce type d’affichage pourrait être étendu à d’autres prescriptions de médicaments. Pour cela, il faudrait séparer et abstraire le code d’affichage des diagrammes actuellement présent dans l’aide AVK, cette “modularisation” permettant de :

I. reprendre le code sur des bases saines (les spécifications sont désormais bien arrêtées) ;
II. faciliter la maintenance du code, la manipulation de fichiers, et l’implémentation de nouvelles fonctionnalités.

Ce travail pourra être réalisé par le DIH, pour le compte de l’hôpital ; ou encore par Médasys S.A., l’industriel en charge du développement de DxCare®. En effet le concept d’aide sous forme d’histogramme a été présenté à la R&D de l'entreprise et celle-ci poursuit des travaux d’études préliminaires, en attendant les résultats des premiers tests de l’aide en production.

2ème Partie : Projet ILOG JRules

Objectif Initial

La seconde mission qui me fut confiée consista à réimplémenter l'aide AVK Javascript, réalisée durant la première partie de mon stage, à l'aide d'un outil de programmation dédié : le moteur de règles *JRules*®, de la société *ILOG*. L'utilisation d'un moteur de règle n'ayant jamais été expérimenté dans le service, mais sérieusement envisagé, ma mission fut de comprendre et d'évaluer cet outil, à en dégager un intérêt éventuel, et à documenter le travail effectué.

Le travail comporta 4 étapes :

1. Découverte des moteurs de règles, État de l'art ;
2. Prise en main / installation / utilisation d'ILOG ;
3. Rédaction d'un "mode d'emploi" condensé pour l'équipe ;
4. Analyse de l'intérêt de cette solution (ILOG) face à la première (Javascript).

Résultats

Voici un extrait de la documentation que j'ai rédigé sur ILOG, débarrassé de ses aspects techniques. Pour une prise en main plus poussée, on peut se référer :

- à la documentation HTML fournie avec ILOG [13] ;
- à la documentation on-line disponible sur le site d'ILOG [14] ;
- ainsi qu'aux tutoriels fournis avec ILOG :
    - **Tutoriel 1 :** Définir un vocabulaire (Notion de projet, de BOM, de XOM, verbalisation d'une classe) ;
    - **Tutoriel 2 :** Définir des règles (Business Rule, Decision tree, Decision Table, Technical rule, Template, catégories) ;
    - **Tutoriel 3 :** Query & reports ;
    - **Tutoriel 4 :** Rulesets & ruleflow (Rule Package, Tasks, interface graphique, algorithmes Rete, FastPass & séquentiel) .

Position du problème

Informatisation d'un secteur d'activité

L'informatisation d'un secteur d'activité qui ne dispose pas encore de système d'information évolué et élaboré se fait en général avec deux types d'outils :

- les outils bureautiques (traitement de texte, messagerie électronique, agenda électronique, etc.) qui couvrent les besoins communs à de nombreux métiers ;
- les logiciels métiers spécifiques (logiciel d'ophtalmologie, de prescription de médicaments, etc.) qui couvrent des besoins plus spécifiques.

Les premiers existent en nombre et sont (relativement) peu coûteux ; on prend en général une solution existante (Microsoft Office, Open Office, etc.). Les logiciels métiers, eux, sont généralement moins nombreux et plus coûteux, car ils doivent être développés au cas par cas. Les programmes ainsi conçus se cantonneront à cette branche d'activité, car trop spécifiques pour être utilisés ailleurs.

Le moteur de règles

Le moteur de règles est un outil qui permet de séparer, dans le code, ce qui est propre au métier (les règles métier) de ce qui relève de la programmation [15]. Les règles métiers peuvent prendre toutes sortes de formes : procédures manuelles (orales ou écrites) de l'entreprise, formules de calcul d'un tableau Excel, code "en dur" dans un logiciel métier, etc. Le savoir et l'expérience de l'entreprise sont ainsi disséminés un peu partout, sous plusieurs formes.

En centralisant les règles dans un moteur de règles, l'expérience acquise par cette entreprise ou ce métier est capitalisée et peut être réutilisée, facilitant ainsi l'informatisation du savoir "métier". Il est alors possible, pour l'utilisateur final (ici le médecin) de créer et modifier les politiques métier dans un langage naturel, sans faire appel à un programmeur pour re-coder le logiciel, ce qui permet d'abaisser la frontière du développement informatique.

Les moteurs de règle au DIH

Dans le cadre de l'HEGP, le DIH envisage depuis près d'un an d'utiliser un moteur de règles pour factoriser les règles récurrentes qui interviennent lors de la prescription, par ex. : "si on a déjà prescrit au patient une sérologie VIH, VHC ou VHB il y a moins de 90 jours, il est a priori inutile de répéter cette prescription". Ou encore dans le cas des AVK : "si l'INR du jour est disponible, la dose d'AVK recommandée est indiquée dans le nomogramme de référence [3], et dépend de plusieurs paramètres (la date de début de traitement, la valeur de l'INR du jour et la dose précédente.)".

Le moteur de règles choisi pour cette étude est le logiciel JRules de la société ILOG, en raison de la proximité de leur support (l'entreprise est française), mais aussi de leur crédibilité dans le secteur (leader dans le domaine des moteurs de règles métiers avec plus de 2500 clients, dont American Express, US DoD, Airbus, BMW, Michelin, Alcatel, SNCF…). De plus , ILOG a été choisi par Medasys pour être intégré à DxCare.

Prise en main de JRules

Structure d'ensemble

JRules est essentiellement constitué de 3 éléments: un élément pour les informaticiens (Rule Studio), un pour les utilisateurs finaux (Rule Team server), et un pour synchroniser les deux (Rule Execution Server). Nous les étudierons ici dans leur version Java, sachant qu'une version .Net existe aussi.

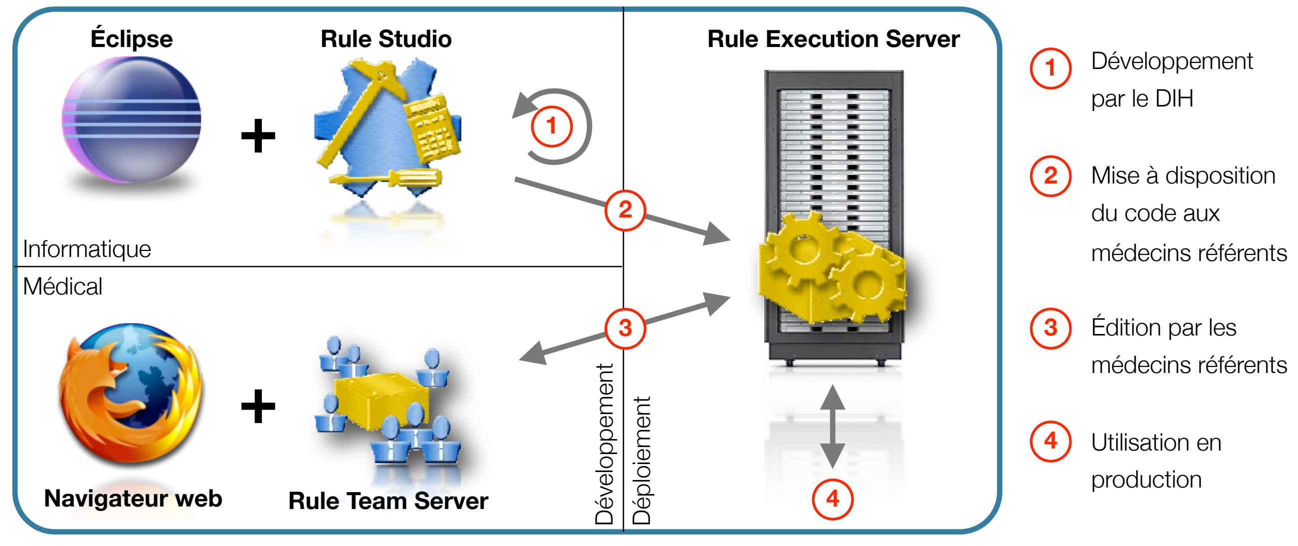


Figure 9 : Cycle de vie d'une règle, illustrant les différents modules de JRules utilisés.

Rule Studio

Rule Studio est un plug-in pour l'environnement de développement Éclipse, et permet au sein de celui-ci de créer les règles et de les exécuter ① en local. On peut ensuite synchroniser, avec Rule Execution Server, le travail effectué ②. L'intégration de Rule Studio au sein d'Éclipse permet d'utiliser un environnement de développement déjà connu, et apporte en plus des interfaces graphiques conviviales pour éditer les règles (tableaux, schémas, diagrammes…).

Rule Team Server

Rule Team Server est une application Java qui permet de centraliser les règles, et qui offre aux utilisateurs finaux une interface web pour l'édition des règles ③ : l'environnement d'édition est plus simple et plus léger, car il s'adresse à un public non informaticien, et permet le travail collaboratif.

Rule Execution Server

ILOG Rule Execution Server est une application qui offre un environnement d'exécution pour faire fonctionner les règles ④, une fois qu'elles ont été créées, aussi bien avec Rule Studio que Rule Team Server. Cette application encapsule le moteur de règles JRules proprement dit, et inclut également une interface web qui permet de gérer l'exécution des règles (par ex. la répartition de la charge), et de récupérer diverses statistiques.

Utilisation

Vue d'ensemble

Le fonctionnement d'ILOG JRules repose sur le modèle MVC : modèle-vue-contrôleur, qui permet de séparer proprement la structure, la présentation et le comportement d'un programme. En programmation Java, on aurait 3 types de fichiers : la classe et ses attributs (modèle), l'interface, et le code proprement dit, composé de branchements, tests, boucles etc. ILOG propose d'encapsuler ces entités : un "BOM" (Business Object Model) pour la classe, et les rules & query pour le comportement. De plus, ILOG propose différentes interfaces pour générer le code correspondant aux "rules" : tableaux, diagrammes, etc.

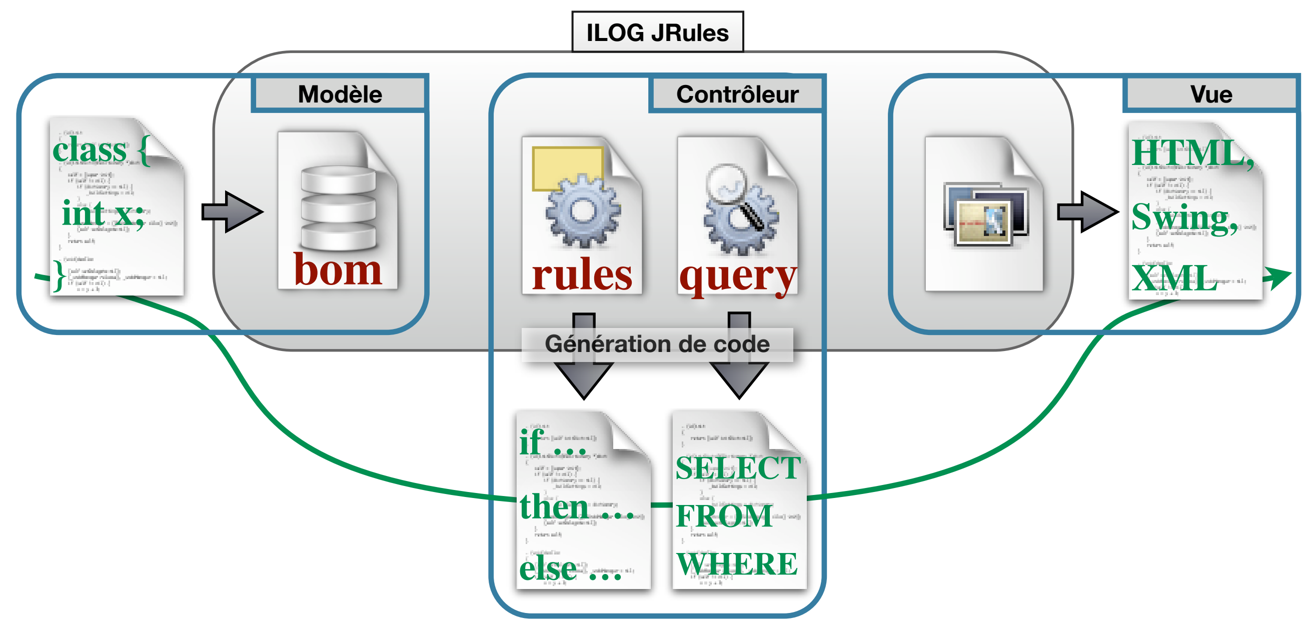


Figure 10 : Vue d'ensemble des différents fichiers générés par ILOG JRules

1 - Modèle



Avant toute chose, il faut créer des classes Java qui définissent les entités qui interviennent dans la règle (Un médicament possède un nom, un numéro OMS, etc…). Ces classes sont essentiellement vides, c'est à dire qu'elles définissent juste la structure (attributs), mais pas le comportement (méthodes), qui lui sera défini grâce aux règles. Une fois cette classe Java créée, on va l'importer dans Rule Studio, et l'encapsuler dans un "BOM". Un BOM est une représentation du modèle Java; c'est là que l'on va "verbaliser" la classe, c'est à dire transcrire, en langage naturel, le nom des classes, des attributs et des méthodes. Il ne s'agit pas uniquement de traduire en français, mais aussi de préciser le genre, le pluriel, l'attribut partitif (de + le = du), etc.

2 - Règles

Pour générer les règles, il existe 4 types d'interfaces :

<table>
<tr><th>Appellation ILOG</th><th>La règle prend la forme d'un …</th><th>Exemple</th></tr>
<tr><td>Business Rule</td><td>Enoncé en langage naturel</td><td>"Si le nom du médicament est Fluindione, alors ..."</td></tr>
<tr><td>Decision Tree</td><td>Diagramme</td><td>customer age
[15,20] → rule 0 : reject 'the current rental agreement'
[21,30] → rule 1 : set the price of the best offer of 'the current rental agreement' t</td></tr>
<tr><td>Decision Table</td><td>Tableau</td><td>
<table>
<tr><th></th><th>State</th><th>Age of the customer Min</th><th>Age of the customer Max</th><th>Rental Rejected</th><th>Reason</th></tr>
<tr><td>0</td><td>Rhode Island</td><td>18</td><td>21</td><td>true</td><td>In Rhode Island sta...</td></tr>
<tr><td>1</td><td></td><td>21</td><td>25</td><td>true</td><td>In Rhode Island sta...</td></tr>
<tr><td>2</td><td>New York</td><td>18</td><td>21</td><td>true</td><td>In New York state,...</td></tr>
<tr><td>3</td><td></td><td>21</td><td>25</td><td>false</td><td></td></tr>
<tr><td>4</td><td>New Hampshire</td><td>18</td><td>21</td><td>true</td><td>In New Hampshire ...</td></tr>
<tr><td>5</td><td></td><td>21</td><td>25</td><td>false</td><td></td></tr>
<tr><td>6</td><td>Massachusetts</td><td>18</td><td>21</td><td>false</td><td></td></tr>
<tr><td>7</td><td></td><td>21</td><td>25</td><td>false</td><td></td></tr>
</table>
</td></tr>
<tr><td>Technical Rule</td><td>Code IRL (ILOG Rule Language) : code "bas niveau" d'ILOG, proche de Java)</td><td>
<pre>
when{?x: Fish(color==yellow;);}
then{
  retract ?x;
  insert Fish(yellow, shark);
}
</pre>
</td></tr>
</table>

Figure 11 : Tableau des différentes formes prises par les règles métier.

Création d'une "Busines Rule" :



Dans l'éditeur de code, on écrit les règles sous forme naturelle, par ex: "Si le nom du médicament est Fluindione, alors …". Lors de la frappe, Rule Studio propose de compléter automatiquement les mots et les phrases, ce qui agit comme un garde-fou : Bien que l'on écrive en langage naturel, on ne peut pas écrire n'importe quoi pour autant. Ce mécanisme est assez puissant, et permet d'écrire rapidement de longues règles. (Rule Studio s'appuie en fait sur le BOM que l'on a créé, et dispose ainsi d'une certaine 'compréhension' du code que l'on écrit. Par exemple, en écrivant "Si le nom…", il propose automatiquement "…du médicament", car il sait que nom (Usual_name) est un attribut de Médicament (Drug.java).).

Création d'un "Decision Tree" ou "Decision Table" :



Ces deux formes de règles s'éditent graphiquement : à l'aide d'outils adaptés, on peut dessiner à la souris un diagramme, ou remplir une table de décision. Les règles sont alors très semblables à celles rencontrées dans la vie courante, *i.e.* en dehors de l'univers de programmation. Ces deux éditeurs ont aussi leur propre garde-fou dans la mesure où les conflits entre règles incompatibles sont détectés et signalés à l'utilisateur.

Création d'une "Technical Rule" :



L'édition de ces règles nécessitent la maîtrise d'IRL (ILOG Rule Language), le code "bas niveau" d'ILOG, proche de Java. Si l'approche est plus complexe (plus proche d'un vrai langage de programmation), les possibilités ouvertes sont en revanche plus grandes.

3 - Rule Templates



Pour faciliter la génération de règles récurrentes, on peut utiliser des modèles de règles : ce sont des règles partiellement écrites, qui permettent de générer beaucoup de règles ayant la même structure. Ce sont ces modèles de règles que l'utilisateur final pourra ensuite compléter une à une, depuis son navigateur Web : pour chaque variable, l'utilisateur a une liste déroulante de valeurs. Il peut ainsi éditer sa règle, suivant la structure du template. Si la variable est une date, un calendrier apparaît pour choisir la date, s'assurant ainsi du bon format de la date rentrée (par ex. JJ/MM/AA ou JJ - MM - AAAA).

4 - Requêtes



Il est possible de spécifier des requêtes permettant de retirer une ou plusieurs règles ; cependant, celles-ci ne portent que sur les règles : les requêtes ne remplacent pas une fouille dans une base de données, et ne se substituent pas à SQL. En revanche, cela permet de retrouver une famille de règles, et d'en dégager des statistiques par exemple.

5 - RuleFlow

Une fois que les règles sont définies (sous forme d'énoncés, de tableaux, de diagrammes…), il faut créer un RuleFlow. Celui-ci est une méta-règle qui définit l'enchaînement entre les règles écrites précédemment. Ce RuleFlow se génère graphiquement, donc simplement. On glisse les règles dans la zone d'édition, puis on les relie par des flèches. Il est aussi possible de créer des embranchements, etc. Ensuite, on définit un point de départ, un point d'arrivée, et le programme peut se lancer.

Évaluation & Perspectives

JRules facilite la création de règles, et en abaisse la difficulté technologique en offrant un environnement de programmation en langage “naturel”, et localisé (*i.e.* en français.); cependant, il ne permet pas de se passer d'un département informatique. La création d'une règle initiale, ou d'un modèle de règle, doit tout d'abord se faire par un informaticien, sous Éclipse : nécessité de formaliser le problème, de créer des classes Java, de les encapsuler, de les déployer. Une fois l'infrastructure mise en place, la modification des règles peut se faire, soit sous Éclipse, soit depuis le navigateur Web.

Il me semble donc que l'emploi d'un moteur de règles est indiqué pour des *familles* de règles, et non des règles uniques, que le médecin pourra créer à partir d'un modèle créé par le DIH. On peut penser comme exemple aux familles de médicaments, ou aux recherches de répétitions inutiles de prescription. L'édition de règles uniques est possible aussi, mais la différence avec une méthode traditionnelle (Javascript par ex.) est moins flagrante.

Le moteur de règles est aussi très pratique dans le cas de règles à *changements fréquents* ; pour une entreprise, cela peut concerner les règles tarifaires ou les offres commerciales ; à l'HEGP, cela peut concerner les règles qui évoluent sur une base mensuelle / annuelle / pluri-annuelle : politique de gestion des lits, politique de coûts, ajustement des pratiques médicales, décisions du PHRC (Programme Hospitalier de Recherche Clinique), etc. Cela peut également s'appliquer aux procédures en phases de test ou de développement par le DIH.

Bilan du Stage

Les deux mois et demi passés au DIH ont pour moi été l'occasion de mettre en pratique les enseignements de l'école, mais aussi d'acquérir de nouvelles compétences. Ces éléments m'ont permis de préciser les perspectives qui s'ouvriront à moi l'année prochaine, à l'issue de ma scolarité.

Apports de l'Entreprise

Ce stage m'a permis d'acquérir de nouvelles compétences, aussi bien techniques qu'organisationnelles :

Compétences Informatiques

- Programmation web dynamique : HTML, CSS, Javascript (et dans une moindre mesure : AJAX, DOM, PHP) ;
- Prise en main d'un moteur de règle (ILOG JRules), très utilisé dans les administrations et les grands groupes ;
- Sensibilisation à l'existence de techniques émergentes prometteuses (AJAX, Ruby on Rails, Programmation agile) ;
- Familiarisation plus poussée avec des outils de développement comme Xcode, Safari, Firefox et leurs plug-in ;
- Apprentissage des "bonnes" pratiques de développement en milieu professionnel : Systèmes de SCM (Source Control Management), tests automatisés, fichier de compilation ("1 click build") .

Compétences en gestion de projet

- Notion d'Assurance Qualité (QA) : Mise en place d'outils statistiques pour évaluer l'impact de la mise en place du projet;
- Capitalisation du travail effectué (élaboration de batteries de test ; rédaction de documentation écrite — plus pérenne que la transmission orale.) ;
- Importance des aspects politiques et économiques d'un projet : les problèmes sont politiques, économiques, puis techniques, dans cet ordre ;
- Écoute de l'utilisateur final : L'élaboration des histogrammes a été l'occasion de réfléchir à la présentation de l'information, à l'ergonomie et à l'écoute de l'utilisateur, puis à transcrire cela en spécifications techniques (ce qui est d'ailleurs l'objectif de la formation MOSI à l'École des Mines) ;

Application des enseignements de l'école

L'aspect le plus formateur de mon stage se résume dans les domaines et les compétences nouvelles énumérées ci-dessus ; néanmoins, je n'aurai pas pu sereinement progresser sans certaines connaissances de base, soit enseignées à l'école, soit acquises en autodidacte, par "curiosité scientifique".

Enseignements de l'École

Sans être un enseignement en lui-même, je dirai que c'est d'abord l'esprit scientifique & la rigueur de l'ingénieur dont il a fallu faire preuve dans ce stage, que ce soit pour formaliser les spécifications des médecins (et envisager tous les cas particuliers) ou pour identifier les bogues. Plus particulièrement, les acquis en termes techniques dispensés à l'École et mis en œuvre pendant le stage furent SQL, Java, et la manipulation d'Éclipse.

En terme de gestion et de management, l'expérience menée dans les associations étudiantes de l'école me fut précieuse : travail en équipe (organisation du Forum Est-Horizon), présentations orales et écrites de qualité (au Forum : pôle PAO (mise en page & travaux d'imprimerie), étude de présentations professionnelles[16][17]).

Perspectives

Les entreprises pour lesquelles j'aimerais travailler seront celles qui auront ces deux composantes : technique et créative. Technique, de par ma formation d'ingénieur et mon intérêt pour les nouvelles technologies; Créative, car je m'intéresse aux domaines qui innovent et qui conçoivent des solutions technologiques audacieuses pour réaliser tout ce qui semble impossible : cinéma, effets spéciaux, animation 3D, automates, parcs à thèmes…

De ces deux aspects, ce stage a renforcé l'aspect technique, avec notamment un accent mis sur des technologies de visualisation et de mise en page web (HTML, CSS, Javascript/DOM), tout en me sensibilisant à la présentation de l'information. Ceci prolonge et complète mon travail de Pôle PAO au Forum Est-Horizon (graphismes, mise en page, impression professionnelle…), ainsi qu'au journal TV de l'école (montage vidéo), offrant ainsi une cohésion dans mon parcours autour des technologies de l'image, qu'elles soient fixes, animées ou imprimées.

D'autre part, ces milieux créatifs (imprimerie, publicité, cinéma) utilisent beaucoup la plateforme Mac OS X, que je maîtrise et pour laquelle j'aimerai programmer professionnellement (programmation Cocoa/Objective-C); aussi une prochaine étape de mon parcours pourrait s'orienter vers ce type d'entreprises.

Conclusion

Ce stage de deuxième année fut très instructif et m'a montré ce qu'était véritablement un travail en milieu professionnel. Travailler dans le domaine médical, de plus sur un prototype à développer et sur un outil d'avant-garde, fut une expérience particulièrement inédite qui m'a ouvert les yeux sur la réalité du secteur. Accéder de près à la réalité du développement informatique, même si cette vision fut tronquée par les nombreuses spécificités de l'hôpital, m'aura permis de réfléchir et de me faire une idée plus précise sur mon orientation future.

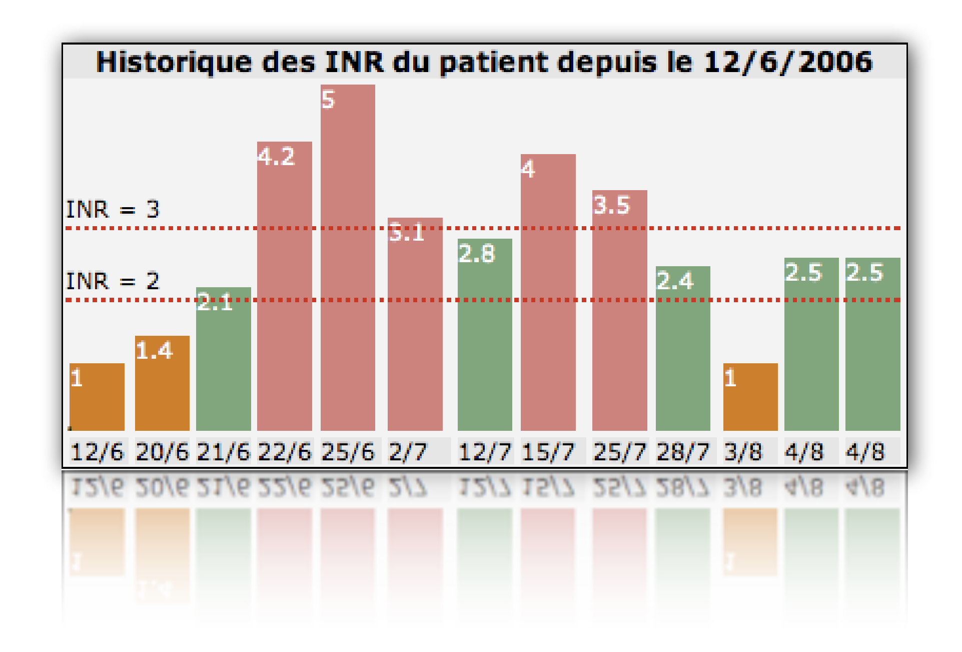
Je ne manquerai pas d'appliquer les enseignements de ce stage l'année prochaine, dans mon cursus à l'ISIA - Mines de Paris, ainsi que tout a long de mon parcours professionnel.





Annexe : Guide de l'Aide AVK

Dimitri Dupuis-Latour — Hôpital Européen Georges Pompidou



Nomogrammes d’ajustement

Nomogramme d’initiation

Jour	INR	Posologie (mg)	Contrôle	numéro
J0	<1.4	20	J+2 (J2)	101
J2	1-1.4	30	J+2 (J4)	121
	1.5-1.7	25	“	122
	1.8-2.3	20	“	123
	2.4-3	15	“	124
	>3	10	“	125
J4	<1.8	+10	J+2 (J6)	141
	1.8-2	+5	“	142
	2.1-2.5	inchangée	“	143
	2.6-3	Si dose > 20 mg alors: –5 mg	“	144
		Si dose <= 20 mg: inchangée	“	145
	3.1-4.5	Si dose > 15 mg alors: –10 mg	“	146
		Si dose <= 15 mg alors: –5 mg	“	147
J6	<2	+5	J+2 (J8) (=>J6)	161
	2-3.1	inchangée	J+7 (J13) (=> table Flc: J0)	162
	3.2-4.5	-5	J+2 (J8) (=>J6)	163

Nomogramme long cours

Jour	INR	Posologie (mg)	Contrôle	numéro
J0	<2	+5 mg	J+7 (J7) (=>J0)	201
	2-3.1	idem dose précédente	J+14 (J14)	202
	3.2-4.5	–5 mg	J+7 (J7) (=>J0)	203
J14	<2	+5	J+7 (J21) (=>J0)	211
	2-3.1	inchangée	J+21 (J35) (=> J14)	212
	3.2-4.5	-5	J+7 (J21) (=>J0)	213

Nomogramme surdosage

Jour	INR	Posologie (mg)	Contrôle	numéro
J0	4.5-9	Ne pas prendre la dose suivante	J+1 (J1)	301
	>9	Sauter la dose suivante et envisager un traitement par vitamine K (lien sur Procédure)	J+1 (J1)	302
J1	< 5	dose = (dernière dose différente de 0) – 5 mg	J+2 (J3)(=> table Flc:J0)	311
	5-9	Ne pas prendre la dose suivante	J+1 (J2)(=>J1)	312
	9-20	Sauter la dose suivante et envisager un traitement par vitamine K (lien sur Procédure)	J+1 (J2)(=>J1)	313

Gestion des situations possibles

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">INR du JOUR</td></tr>
<tr><td>OUI</td><td>NON</td></tr>
<tr><td rowspan="2">JT</td><td>Dans Nomog</td><td>1) <u>Situation standard:</u>
Messages (selon JT et INR), avec Dose et date de contrôle recommandés</td><td>2) Message suivant : « Attention ! Le résultat de l’INR du jour n’est pas disponible. Vérifier que ce test a bien été effectué avant de prescrire votre AVK »</td></tr>
<tr><td>Hors Nomog</td><td>3) Situation où INR prescrit hors protocole
Selon INR
A) Si > 4.5 => boucle sur surdosage
B) Si < 4.5 voir Nomog :
a) Si Ic :
On décale tout => Situation 1 (on s’assure que le patient reste 14 jours à la même posologie avant de contrôler tous les 3 semaines)
b) Si init :
Message suivant : « Contrôle non prévu ce jour par le protocole ! Aucun conseil d’ajustement de la dose ne sera donné : si vous décidez de modifier la posologie aucune aide à la prescription ne sera disponible ultérieurement pour ce patient, sinon nous vous invitons à re-contrôler l’INR le … (date du contrôle prévu au dernier message d’aide) »</td><td>4) <u>Situation de prescription intermédiaire</u>
Rappel du dernier message (dernières dose et date de contrôle stockées)</td></tr>
</table>

Détail du Javascript

1) CSS : définition du style

2) Définition des variables

a) Variables du questionnaire

var fourchette; getFourchette(); La fourchette d'INR cible, vaut 0(= 2 a 3) ou 1(= 3 a 4.5)
var nomog; getNomog(); 1 = NomogInit, 2 = NomogLongCours, 3 = NomogSurdose
var J0;getJ0();
var dose; getDoseEff(); dose conseillee, en mg

b) Variables du map

var INR; valeur numérique de l'INR
var labelINR ; manière d'afficher l'INR getINR();
var date_INR; getDate_INR();
var dosePrescrite; getDosePrecedante() ce qui a réellement été administré au patient

c) Variable de l'URL

var NIP ; recupNIP(); Pour pouvoir retrouver le patient dans la BDD

d) Variables internes au code

var index = 1; nombre d'affichages pour ce patient
var inclusion ;
var dateControle; La date du prochain Contrôle INR
var message = ""; Le Message de sortie
var JT = null; JT n'est pas une date, mais un nombre de jour calcule dynamiquement
var dateDuJour=new Date();
var DoseNonNulle = 0;
var xmlhttp = getHTTPObject(); c'est un XMLHTTPRequest permettant de communiquer avec la base de données
var cheminPHP = "http://idol3/aide_prescription/AideAVK_(prod)/Aide-PHP/"; chemin d'acces au dossier des scripts PHP
var results_query;

e) Variables pour affichage du diagramme

var xMax; nbre total de jours
var xMin; nbre de jours du plus petit intervalle
var echelle; nbre de jours par pixels
var longueurTotaleEnPx; xmax * echelle
var yMax;

3) Programme

```
<body>
<Script type="text/javascript">testNIP();</Script>
<div id="MESS" class="message"></div>
<div id="DIAGRAMME"></div>
<div id="FourchetteDiagramme"></div>
<div id="nomogrammes"></div>
</body>
```

`testNIP()`

Connexion avec la base de données et récupération des données liées au NIP si il existe dans la base -> `analyserNIP()`

`analyserNIP()`

Si NIP existe déjà on utilise les variables stockées -> `utiliserVariablesDeLaBase();`

Sinon on prend celles du questionnaire -> `utiliserVariablesDuQuestionnaire();`

```
    genererLeMessage();
    ecrireDanslaBase();
    GetHistorique();
    afficherLeMessage();
    afficherNomogramme();
```

4) Fonctions dans l'ordre chronologique d'appel

a) les variables

`utiliserVariablesDeLaBase()`

Le PHP renvoie les informations sous la forme d'une chaine ex : 123,oui,0,2006/7/23,1.8,1,22,2006/7/25,3,22 et les variables sont remplies à partir de celle-ci. `index ++` pour gérer l'historique des affichages

`utiliserVariablesDuQuestionnaire()`

La première fois que l'aide se déclenche pour un patient, on utilise les variables du questionnaire. `index = 1;`

b) Calcul de l'aide

`genererLeMessage()`

Elimininination de tous les cas particuliers puis -> `aideFluinidione()`

`aideFluinidione()`

Préparation avant d'entrer dans le cœur du programme

```
    CalculerJT();
    JTdansNomog();
    INRdispo();
    TestSurdosage();
    CalculerDoseEtDate();
```

`CalculerJT()`

Date du jour moins J0

`JTdansNomog()`

Initiation : 0, 2, 4 ou 6 ; Long court : 0 ou 14 ; Surdosage : 0 ou 1

`INRdispo()`

Teste si l'INR du jour est disponible, et et lui affecte sa valeur si elle existe. DxCare pas d'INR : INR = "######"

`TestSurdosage()`

Vérifie s'il y a surdosage, *i.e.* INR > 4.5 -> `CalculerJT()`

`CalculerDoseEtDate()` //Le cœur du programme

```
    switch (nomog) {Les 3 nomog sous forme informatique}
        JPlus();
        CreerMessage();
        J0Avance();
        messageSurdosageZero();
        messageSurdosageZeroVitaminesK();
```

`JPlus()`

Retourne la date qu'il sera i jours plus tard

`CreerMessage()`

Construction du conseil personnalisé

`J0Avance()`

Avance J0 de i jours

`messageSurdosageZero()`

Cette fonction affiche une dose de zéro -> `dateControleFR()`

`messageSurdosageZeroVitaminesK()`

Cette fonction affiche une dose de zéro + un lien vers le traitement Vitamine K : `ouvrirPopUp()- dateControleFR();`

`dateControleFR()`

Met les dates en format français

`ouvrirPopUp()`

Permet l'affichage de la procédure HAGP stockée sur idol dans une pop-up

c) stockage des données calculées

`ecrireDanslaBase()`

Ecrit dans la base les 6 variables susceptibles d'avoir change et les 2 traces (+ le NIP et l'index)

d) récupération historique des INR et doses AVK

`GetHistorique()`

Connexion avec la base et récupération des données liées au NIP si il existe dans la base -> `traiterHistorique()`

`traiterHistorique()`

On met les valeurs dans un tableau : Date, Label INR et Dose ; On a alors en mémoire le tableau suivant (NB : ce tableau est affichable en décommentant la fonction voirTableau(array), ce qui est très pratique pour le débogguage.)

On calcule xMax et yMax du graph

`voirDiagramme()`

`fermerLaFenetre()`

INR	Date	Ecart (en jours)
1	2006/6/12	8
1.4	2006/6/20	1
2.1	2006/6/21	1
4.2	2006/6/22	3
5	2006/6/25	7
3.1	2006/7/2	10
2.8	2006/7/12	3
4	2006/7/15	10
3.5	2006/7/25	3
2.4	2006/7/28	6
1	2006/8/3	1
2.5	2006/8/4	0
2.5	2006/8/4	0
2.5	2006/8/4	0

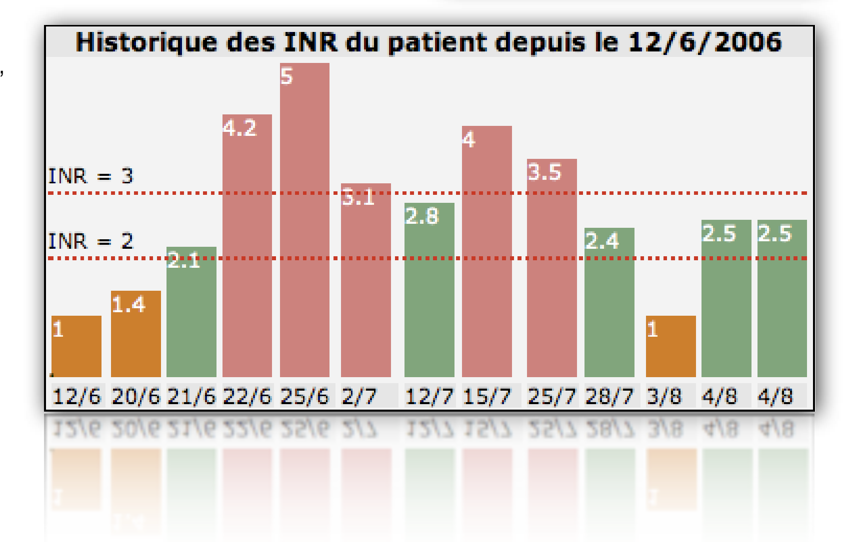
`voirDiagramme()`

Création de la fenêtre, du tableau, du caption, du body, des dates & doses, les guides.

Le diagramme est constitué de 2 tableaux HTML superposés:

- un pour les barres verticales
- un par-dessus avec les guides de la fourchette INR.

Chacun de ces tableaux est dans une balise <div>, ce qui permet de les positionner n'importe où, et en particulier l'un au dessus de l'autre. L'aspect est contrôlé par CSS.





`fermerLaFenetre()`

Ferme la fenetre… qui l'eut crû ?

e) affichage de l'aide individualisée

`afficherLeMessage()`

Div, table, body, tr & td imbriqués les uns aux autres ; permet d'afficher le conseil, le lien vers nomogramme et le lien pour fermer la fenêtre

f) affichage du nomogramme

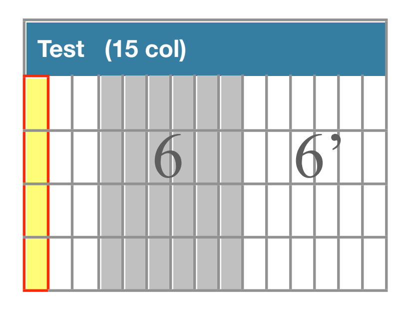
`afficherNomogramme()`

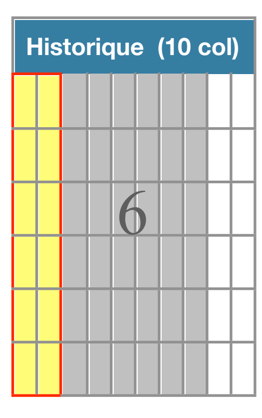
Servait quand on affichait nomogramme dans la même fenêtre que le conseil

Détail de la Base de Tests

Vue d'ensemble

On utilise 2 tables : test et historique. Chacune contient les 6 variables susceptibles d'être modifiées (les cases grisées) Dans le schéma suivant, le cadre rouge correspond à la (les) clé primaire de la table.



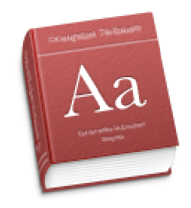


- Table test : 1 unique ligne par patient, et on n'écrase pas les valeurs (pour ne pas corrompre la base de tests). C'est pourquoi les résultats sont stockés dans la colonne 6'.
- Table historique : pls lignes par patient, et on n'écrase pas les valeurs (pour garder une trace de tout ce qui s'est passé). On a donc besoin d'une seconde clé primaire : indexH

Variables

Nom	Type	Description	Commentaire	test	histo
NIP	String	NIP du patient		✔	✔
IndexH	Int	Nombre d'affichages pour ce patient	Incrémenté à chaque affichage		✔
Fourchette	Int	La fourchette d'INR cible: vaut 0(= 2 à 3) ou 1(= 3 à 4.5)		✔	
DosePrescrite	Int	Ce qui a réellement été administré		✔	
Inclusion	String	oui, non ou null (en minuscules)	Modifié si exclus de l'étude	✔	✔
J0	String		Modifié si changement de nomog	✔	✔
INR	Float		Modifié (Mis à zéro) après chaque aide ?	✔	✔
Nomog	Int	1:NomogInit, 2:NomogLongCours, 3:NomogSurdose	Modifié si changement de nomog	✔	✔
Dose	Int	Dose conseillée, en mg	Modifié après l'aide	✔	✔
Date	String	Date du prochain Contrôle INR	Modifié après l'aide	✔	✔
TraceINR	Float	L'INR du jour	Permet de construire le diagramme		✔
TraceDDJ	String	La date ou l'aide a été générée	Permet de construire le diagramme		✔

Bibliographie



Publications Médicales

[1] Degoulet, P., et al., The HEGP component-based clinical information system. Int J Med Inform, 2003. 69(2-3): p. 115-26.

[2] AFSSAPS : Agence Française de Sécurité Sanitaire des Produits de Santé. http://agmed.sante.gouv.fr/htm/5/5710c.htm

[3] Protocole du Pr Jean Noël Fiessinger : traitement anticoagulant. In : Elias A ; 1995 : 122-32

Documentation Technique

Référence

[4] HTML 4.01 : http://www.w3.org/TR/html4/

[5] CSS 2.1 : http://www.w3.org/TR/CSS21/

[6] Apprentissage de Javascript : http://www.w3schools.com/

[7] Quirksmode : liste les incompatibilités HTML, CSS et Javascript entre navigateurs. http://www.quirksmode.org/

Validateur

[8] HTML local : HTML Tidy (Plug-in pour Safari & Firefox) http://www.zappatic.net/safaritidy/

[9] HTML en ligne : http://validator.w3.org/

[10] CSS en ligne : http://jigsaw.w3.org/css-validator/

Débogeur

[11] HTML & CSS :

- Safari (Web Inspector) : http://webkit.opendarwin.org/blog/?p=41
- Firefox (Inspecteur DOM) : http://www.mozilla.org/projects/inspector/

[12] Javascript :

- Safari (Drosera) : http://webkit.opendarwin.org/blog/?p=61
- Firefox (Venkman) : http://www.mozilla.org/projects/venkman/

Moteur de règles

[13] Documentation ILOG off-line : ILOG JRules 6.1/doc/html/index.html

[14] Documentation ILOG on-line (nécessite de créer un compte myILOG) : https://support.ilog.com/private/products

[15] Moteurs de règles, principes généraux : http://en.wikipedia.org/wiki/Business_rules_approach

L'Art des présentations (Keynote - Powerpoint)

[16] Conception de présentations professionnelles : http://www.presentationzen.com/

[17] Soigner ses présentations : http://www.macworld.com/2006/07/secrets/augustcreate/index.php